I.J. Mathematical Sciences and Computing, **2017, 4, 8-19**
Published Online November 2017 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijmsc.2017.04.02

Available online at **http://www.mecs-press.net/ijmsc**

A Multi-view Comparison of Various Metaheuristic and Soft Computing Algorithms

Abdulrahman Ahmed Bobakr Baqais[a*]

[a]*Dhahran, Saudi Arabia*

Received: 08 January 2017; Accepted: 16 March 2017; Published: 08 November 2017

Abstract

AI algorithms have been applied in a wide spectrum of articles across different domains with great success in finding solutions. There is an increasing trend of applying these techniques on newer problems. However, the numerous numbers of algorithms that are classified as AI algorithm hinder the ability of any researcher to select which algorithm is suitable for his problem. The invention of new algorithms increases the difficulty for researchers to be updated about AI algorithms. This paper is intended to provide a multi-facet comparison between various AI algorithms in order to aid researchers in understanding the differences between some of the popular algorithms and select the suitable candidate for their problems.

Index Terms: Metaheuristics, Review, Comparison.

1. Introduction

Artificial Intelligence algorithms brought many advantages to the research literature in any knowledge domain. The ability of the computer to learn reason and decide saves millions of dollars for companies to replace experts. Artificial Intelligence techniques revive in the last decade due to the advances in term of power and memory of today computers enable these algorithms to work fast and return results in a significant short time.

Artificial Intelligence algorithms are broad in nature and there are divided into many categories with no clear cut or borders. In this paper, however, we are going to focus on three classes of AI algorithms namely: Metaheuristics, Soft Computing and Machine Learning techniques.

Many reviews and articles have been published up to date about the various algorithms of metaheuristics. However, this research differs in many aspects: first, with serving the various algorithms, we added a

* Corresponding author. Tel.:+966-13-350-5600
E-mail address: phd_abdulrahman@yahoo.com

philosophy section for each algorithm. The purpose is to facilitate the creation of new metaheuristic by showing how each metaheuristic have a different philosophy in addressing solutions. Second, we build a framework of comparison based on some criteria and evaluate the metaheuristics. This aids researchers and practitioners in selecting the right algorithms according to their preferences or problem types. Third, this review will help the readers to find various references of different AI algorithms in one place.

The term "Metaheuristic" emerged by Glover in 1986 [1]. Since then, the name becomes very popular and has been applied extensively in research. It seems there is no consentient definition of the word metaheuristics. In [2], the authors provided three definitions of metaheuristics where each one is focusing on one aspect or one perspective. They asserted that no precise definition of the word metaheuristic is given in the literature.

The Handbook of Metaheuristics defines it as [3] " in their original definition, are solution methods that orchestrate an interaction between local improvement procedures and higher level strategies to create a process capable of escaping from local optima and performing a robust search of a solution space". Tabli in his book however defines it [4]: "Metaheuristics as an upper level general methodologies(templates) that can be used as guiding strategies in designing underlying heuristics to solve specie optimization problems". This is the more relevant definition to the work discussed here and as such we will adopt it.

Metaheuristic is a set of algorithms that search for an optimal solutions based on two main forces: Intensification and Diversification [5]. Their intrinsic behavior is usually imported from nature or biological observations such as: Genetic Algorithms and Ant Colony Optimization. They have been applied extensively with great success in various domains. We are giving examples of three algorithms in the literature review namely: Genetic Algorithms, Ant Colony Optimization and Tabu Search.

Metaheuristic algorithms are preferred over classical approaches due to their speed, computation cost, and absence of domain knowledge. In addition, metaheuristics shine for non-differential [6], highly modal[7], highly dimensional, strong epistasis problems. The lure of metaheuristics spans several domains, extended to a wide variety of problems.

Even though metaheuristics techniques have been applied extensively in various domains with great success, there are some kinds of problems that metaheuristics can't solve efficiently. The question of failure is still open and the extensive articles addressing this issue make it more clouded..

Why some metaheuristics fail? The reasons vary. But let us define failure in this context. Failure means that the algorithm doesn't meet the results demanded by the implementer. For example, if the algorithm works but gives lower value than the threshold set by the designer, it's considered a failure. If the algorithm provides an optimal solution in a prohibitive time or resources, it's considered a failure. If the algorithm provides an optimal solution in a reasonable time but the solution quality is very bad in comparison to other metaheuristics, it's considered a failure. If the algorithm performance is the same as a random search, it's considered also a failure!

So before explaining the reasons why some metaheuristics fail, we must understand the context where they are applied. In addition, we must define the metrics that drive them to fail. Nevertheless, if the algorithm fails in obtaining the anticipated result based on lack of programming skills of the designer, then it's considered a failure. Since it's difficult to measure the skill level of programming design in different articles, any article declares a failure of an algorithm, it's assumed to be failed unless stated otherwise.

In short, if a metaheuristics algorithm failed in a certain problem, that problem is considered hard.

2. Philosophies of some AI algorithms

2.1. Genetic Algorithm

If the solution can be represented as a set of features, and each feature have an optimum value, then promote exchanging of these optimum features, presumably the one with maximum optimum features presents the optimum solution.

(New solutions are based on recombination of previous solutions)

Genetic Algorithms can be used in the following cases:

- Multi optimal solutions, Evolving helps.
- Combinatorial.
- Solutions can be encoded into separate features (Blocks).
- If Blocks are independent, then optimizing each block is preferable.
- Due to premature convergence, they are appropriate for tactical problems. That is, problems that need good enough solutions (not the ultimate optimum) in a quick time.

GA might perform poorly in the following cases:

1. Continuous problems: mutation and crossover is not efficient.

If I have two solutions 11000110 and 00000101 encoded in discrete GA, and the optimum solution is 11000101, then crossover will allow for that by exchanging half of the first chromosome (string) with the second one. If I have a final solution of 01001110 and the optimum solution is 01111110 then by mutating two adjacent bits (third and fourth bits from the left), I can reach it.

However in continuous GA, if I have two solutions 54.2 and 53.45 and the optimum is 57.9 then designing crossover or mutation to capture the essential information of the solutions and exchange it or invert it is not clear.

2. Problems with less important solutions at the beginning:

GA pushes good solutions to next generation with high emphasis, and slowly discards solutions with weak features in total. This leads to premature convergence. If GA is allowing some deterioration of solutions for some generations, they might assist in finding the optimum.

3. GA is less efficient if there is high correlation between the blocks (epistasis)[8], since it optimizes each block separately to reach for optimum solutions.

2.2. Ant Colony Optimization

In this algorithm, many solutions are constructed individually. A central layer is updated with information from these solutions. New solutions are constructed based on the central layer. In each iteration, this central layer provides control and communication to each set of the solutions.

Unlike GA, multi solutions in ACO are exchanging information not by themselves, but based on a central layer.

(Solutions are constructed in each iteration based on a central layer of information and feedback)

ACO performs well in the following cases:

- Solutions can work as agents and able to exchange information indirectly through the pheromone.
- Combinatorial.
- If the solution path can be divided into different components.
- Availability of heuristic information.

However, ACO might not perform always well in continuous optimization.

2.3. Tabu Search

Generate new solutions by applying modification operators to the current solutions and use memory to ensure not going back to the same state.

Unlike GA and ACO, Tabu search allows for some deteriorate solutions to be kept in the memory for enough time to be explored later [9].

(Solutions are constructed based on modification of current solutions)

Particle Swarm Optimization: bring each individual closer to the best individual.

SA philosophy is similar to Tabu search. Allows some deterioration for number of times. The difference is how far you allow deterioration. Tabu search using a memory list, SA uses a mathematical equation.

2.4. Soft Computing

Soft Computing is intended mainly for problems that doesn’t require an exact solution. Instead, any solution ranges within the threshold is acceptable. These techniques are very helpful when they are uncertainty involved in input collection, data measurement or output calculation. Two major algorithms are discussed in the literature review: Fuzzy Logic and Neural Network.

2.4.1 Neural Network

In each iteration, new solutions are calculated. The construction of new solutions is based on a central layer that updates the information. It acts similarly conceptually to ACO but it differs in the lack of control & communication between different solutions in the same iteration.

(New solutions are constructed based on the feedback of previous solutions)

Neural Network is good in the following cases:

- Approximation as it’s proved to be universal approximator [10].
- They are suitable for Design problems. That is a problem that seeks the optimum approximation without a bounded time.

Nevertheless, number of hidden layer units is getting computationally inefficient when problem dimensionality increases.

2.4.2. Fuzzy Logic

- Is used for elements that have different degree of memberships.
- It’s efficient in uncertainty environments or for noisy input.

However, the model can be complex sometimes and it requires a deep mathematical understanding.

2.5. Machine Learning

Machine learning is a set of techniques that enables the computer to learn; hence the name. They are intended to infer some information or knowledge from raw data. They mainly involve three categories: Clustering, Classification and Association. In the literature review, we opt to discuss only clustering algorithms since they are applied frequently in refactoring.

2.5.1 Clustering

- Clustering is used to group elements based on their degrees of similarity. There are various algorithms that satisfy this purpose. A popular algorithm is k-nearest neighbor.

- Clustering is used when there is no clear division among groups.
- Clustering is not efficient if the similarity between elements is very high.
- Clustering is not efficient if a characteristic is absent from some elements.
- It works the best on numerical values.
- It suffers when the number of elements is very large.
- It's not efficient on high-dimensionality elements with many attributes.
- It's hard to determine how many clusters should be there and how many elements are within each cluster.
- Cluster is used on multi-objective problems.
- Cluster works when the element features have defined values. Otherwise, fuzzy clustering could be used.

3. Descriptions of AI Algorithms

3.1. Genetic Algorithms:

Genetic Algorithm was first mentioned by Holland in his book "Adaptation in Natural and Artificial Systems". Later, the theory of genetic algorithms was discussed, elaborated with series of examples on search and optimization problems by Goldberg in his famous book "Genetic Algorithm in Search, Optimization and Machine learning" which was released in 1989. Since then, various books, conferences and articles discussed the applicability of GA to solve various problems in various domains.

Genetic Algorithms is a subset of a larger class of algorithms named Evolutionary algorithms where the basic theme of these algorithms is to mimic biological nature by evolving the current population in an iterative manner to produce better solutions in each subsequent generation. GA works on random initial solutions (called chromosome) that is composed of variable-size blocks. At each iteration, these blocks are either mixed or flipped using some operators like crossover and mutation to produce new chromosomes with higher values calculated by an objective function called "fitness".

Various implementations of GA operators with many variations, extensions and enhancements have been suggested, developed and applied in the literature. As such, GA is considered as a primary candidate of comparison for any new developed metaheuristic algorithm or any novel solution to unexplored problem.

3.2. Genetic Programming:

Genetic programming is another type of evolutionary algorithms. The idea of genetic programming is to randomly create various programs representing the populations. Each program is represented as a syntax tree of variables, constant and operators. Similarly to GA, at each generation, these programs are evolved to produce better individuals by applying crossover and mutation and then calculating their fitness values.

3.3. Tabu Search

Tabu search was developed by Glover in 1986 [1]; the first used the term "metaheuristic" in his classic paper "Future Paths for Integer Programming and Links to Artificial Intelligence". Tabu search is a search algorithm that was created for the purpose of escaping local minima and exploring new paths during the search journey. Tabu search starts from a particular solution and check its neighbors. Potential neighbors that might contribute to the global optimality path are saved in a" Tabu list". This list works as a memory to prevent the Tabu list from reversing to a poor solution and it strengthen the search by providing more solutions to explore in a case of a local minima trap.

Two features distinguish Tabu lists from other metaheuristics: memory and non-randomization. Glover in his paper argued that Tabu search strategy is not in favor of a good solution that is found by an accidental randomness of the algorithm.

3.4. Hill Climbing

Hill Climbing or greedy algorithm is the simplest algorithm for optimization problems. The algorithm works in a greedy manner by moving from one point to another if the latter is optimized than the former according to some objective function. If all neighborhoods of a current solution are inferior, then the algorithm is terminated. This leads to a situation referred in the optimization literature as “local optima”.

Hill Climbing is not intended to find global optimality in a solution space since it’s not equipped to escape the local optima. However, its strengths in exploitation (local search) enables it to be a primer candidate in a hybrid metaheuristic algorithm.

Different variations, enhancements and adjustments of the algorithm is proposed in the literature to address various problems.

3.5. Ant Colony Optimization

Ant Colony Optimization is an instance of a larger class of algorithms named Swarm Intelligence. The algorithm was a result of a PhD Dissertation proposed by Dorigo in 1982 [11]. The basic idea of this algorithm is to mimic ants behavior when searching for food and apply it on problems where the objective is to find a shorter path between two nodes. Ants travel in random paths during their search for food. Each ant leaves some pheromone that evaporates after a specific time. The most desirable road will attract more ants and it will be marked by many pheromones left over. Hence, this is will be an indicator of the optimal path between nest and food source.

ACO were successfully applied in various problems. As with other metaheuristics, various proposals of extensions, variation or adjustments have been discussed in the literature. Many algorithms were developed in the same way by observing animal behavior in different situations. Bee colony, bat algorithm, firefly and cuckoo search are few to mention.

3.6. Neural Network

NN is a network of connected nodes that mimics the biological neural networks. That’s why it is called Artificial Neural Networks (ANN). It connects inputs to outputs through one or more layers where each layer has one or more nodes. Different models of ANN exist such as: Feed Forward and Back Propagation.

3.7. Particle Swarm Optimization

PSO is developed in 1995 by [12] . It’s a population-based algorithm where many solutions are randomly initialized in the solution space. Each individual, aka solution, point, node...etc., is determined by its position and velocity. In each iteration, each solution updates its two parameters: position and velocity. Moreover, each solution remembers its pest position and the global best position that is found by any of the other solutions and it updates its position to be toward the global best solution. Thus, solutions are getting closer and closer from each other, and after few iterations, most of the solutions will target a narrow area in the solution space that is believed to obtain the global optima.

The applicability of PSO in various problems has been discussed in several articles. Many variations, adaptation and adjustment of the parameters have been proposed in the literature to fit different problems.

3.8. Simulated Annealing

Simulating Annealing algorithm is one of the most implemented metaheuristic in the literature due to its efficiency and simplicity. It’s similar to hill climbing in which it starts randomly with a single point and it’s compared with points in the neighborhood. If the new point is better than the old one, then the algorithm keeps the new point. If it's worse, then the algorithm will keep the new one depending on a threshold parameter t. T starts very big in the beginning allowing for weaker solutions to be kept by the algorithm and it's getting decreased gradually in order for the algorithm to converge.

Simulated Annealing have been applied in various problems and various variants, enhancements or adaptation of the operators have been proposed in the literature.

4. Multi-View Comparison

Metaheuristics differ in various aspects including their operators, objective functions, purpose..etc. This section provides a multi-view comparison based on different criteria on how metaheuristics differ from each other. This is the first step toward deciding which algorithm is suitable to your problem.

4.1. Algorithm Component View:

Table 1. Metaheuristics Summary

Technique	**Parameters**	**Selection**	**Inspiration**	**Memory**	**Stochastic/ Deterministic**	**Single/ population based**	**Greedy/ iterative**	**Evolutionary/non evolutionary**
GA	Crossover and mutation	selection	Nature-Inspired	Memory less	Stochastic	Population-based	iterative	Evolutionary
PSO	Position and speed	Gbest and pbest	Nature-inspired	Memory less	Stochastic	Population-based	Iterative	Evolutionary
Simulated Annealing	Cooling factor and acceptance probability	-	Physical-Inspiration	Memory less	Stochastic	Single-based	iterative	Non-evolutionary
ACO	Pheromone evaporation and pheromone update	-	Nature-Inspired	Use memory	Stochastic	Population-based	Iterative	Non-evolutionary
TS	Memory size	-	-	Use memory	Stochastic	Single-based	Iterative	Non-evolutionary
HC	-	-	-	Memory less	Stochastic	Single-based	Greedy	Non-evolutionary
Bee algorithm	limit, maximum cycle number	selection	Nature-inspired	memory less	Stochastic	Population-based	iterative	Non-evolutionary

This table summarizes the major aspects of these AI techniques. The purpose of this table is two folds: to unleash the components of the metaheuristic that drive them to succeed and to show glimpses of the missing aspects of each metaheuristic that perhaps if were combined together in one algorithm, it could boost the algorithm performance.

Table 2. Metaheuristics Strengths and Weaknesses

Technique	Strength	Weaknesses
GA	Simple. Versatile for large scale problem. Successful in many domains and rich in applications Considered as a baseline approach [6]. Works well for multi modal because of resolvability Robust [13]. They are appropriate for large scale problems due to the fact that multi-dimensionality will not affect the number of function evaluation. Evolutionary techniques such as GA are good for exploring interesting regions. For problems where objective function is hard to be formulated [14].	Slow in convergence around the optimum points [15]. Might visit the node twice or more. No theory of convergence Number of fitness evaluations Scales badly with large initial populations. Information is lost in search Weak solutions are discarded early. Initialization step doesn't fit to all problems. Attempts to optimize on higher terms in equations, rather than optimization the whole equation [16]. Not good for inseparable function, since GA works better with blocks. [17]
PSO	Parameter estimation Less time & memory than GA	Might visit the node twice ore more. No convergence theory Homogenization [15]
ACO	Doesn't visit a node twice. Perform better for distributed problem (component-wise) Positive feedback	No convergence theory
Tabu Search	Uses memory to avoid stuck in local optima twice.	No convergence theory. Tendency to leave local optima causes difficulty in convergence.
Fuzzy	Universal approximator [10]	Number of rules grows exponentially for increased accuracy [10] Suffers from existential theorem [18]. Difficulty in interpreting the results of defuzzification process [19].
Neural Network	Universal approximator [10] Good for high dimensions with fewer examples.	Number of rules grows exponentially for increased accuracy Suffers from difficult internal Representation [20] No reproducibility [20] Insufficient generality with under sampling [20] Suffers from existential theorem [18]. ANN requires a large dataset[18]. Not god for underestimate approximation [21].
SVM	No local minima in learning [7] Error doesn't depend on dimension [7] Robust with fewer examples Handles outliers	
Bee Algorithm	Bee Algorithm performs very well even its parameters are not tuned [22].	
SA	Easy to code [23]. Few parameters Applied extensively in the literature with good results. Proof of convergence exist [24].	Its results are lower than the advanced algorithms Such as: GA and cuckoo search The cooling factor has a strong impact on the results as reported in [25].
Clustering	Rich set of algorithms and variants [26] Wide application [27].	- Scales badly with large data - Number of clusters must be predefined [28] - Sensitive to outliers - High dimensional data [26]

If the epistasis or the dependence between variables are very high, then random search is suitable, if it’s very low, hill climbing will be the most effective, in between of these two extremes, genetic algorithm may perform very well [8]. Larger population increases cost and not necessary increases performance and 200-250 is sufficient. Crossover percentage from 70-75% with a mutation of 0 to 0.02 leads to a significant improvement [29]. In GA, the population size correlates with the fitness landscape [13]. Ants and bees algorithm shouldn’t be the first choice in continuous optimization [30]. Takagi-Sugeno-Kang is better than Mamdani for function approximation [31].

In this view, different criteria are defined with a brief description. All algorithms will be addressed by these criteria. These criteria are selected carefully as to be helpful for researchers to decide on which algorithm to select for their problem.

Table 3. Comparison Criteria

Criteria	metaheuristics
Transparency	GA & PSO are transparent.
Scalability	GA scales badly with number of population. NN scales badly with hidden neurons [10]. Fuzzy logic scales badly with fuzzy rules [10].
Purpose	GA for general purposes [6]. Evolutionary optimization algorithm is effective in problems where it’s very hard to formulate an objective function [14]. PSO for parameter estimation [32]. NN for handling outliers [31]. Fuzzy for handling imprecision.
Optimization Determination	Not Activation function in NN , but the topology. Not the parameters of Fuzzy logic, but the rules. Not the population of GA, but the operators.
Measuring function	Changing RMSE , Relative absolute or absolute gives us similar performance.
Lerning Curve	HC and Tabu search is at level 1 SA is at level 2 GA and PSO at level 3 ACO and ABC at level 4 Fuzzy Logic at level 4
Learnability	GA and NN, SA due to the abandon of literature, applications, tools. PSO and ACO at level 2 Fuzzy Logic and ABC at level 3
Customizable	HC , SA , GA and PSO at level 1 Fuzzy logic and Neural Network at level 2
Convergence Analysis	in GA Squared E converges quickly than Absolute Error , though both of them converges to the same level[16] SA is proved to converge to the optimal solution by using the right cooling [24]. ACO is proofed to be converged to the optimal solution [33]. Tabu search convergence in finite steps is discussed in [34]. A claim of Artificial Bee Algorithm convergence to the global is presented in [35]. No proved convergence rate found in PSO [36]
Others	GA adaptation from generation to generation PSO adaptation in a generation [37] GA uses crossover and mutation, PSO uses a mathematical systematic approach [37]. GA is robust for curve fitting [16]. GA produce an offspring after considering two parents vectors while other algorithms such Harmony Search considers all parent vectors [38].

Different criteria have been listed in Table 3 to compare between different of these algorithms. Not all algorithms are tested in each criterion. These criteria differ in their purpose and importance for each designer. Some designers would like a fast convergence. Others may focus on the ease of the algorithm to be learnt.

5. Popularity- View

Using Google Trends, we searched for different metaheuristics in order to capture the interest of these algorithms over time. Google Trend only allows for 5 terms to be compared, so we used the most popular metaheuristics which are: SA, GA, PSO, TS and ACO. Even though Google Trend is a basic tool and doesn't provide use with various facilities to run perfect comparison, the obtained results is very helpful to give us a glimpse about the popularity between these metaheuristics.

GA was on the top and considerably away from competition from any of the other metaheuristics. GA's simplicity and availability of tools and variations contribute mostly to this result in our opinion. Simulated Annealing scored the second in average across the years, however, in recent years, other algorithms such as PSO is more popular and appealing to the researchers. Ant colony optimization has a good score in average and ranked better than Tabu search which is seems fading across the years.

We believe that the analysis returned by the tool is short and brief and doesn't constitute for an affirmative conclusion. However, one obvious point is shown that GA is still popular and attracting researchers regardless of the various metaheuristic algorithms proposed by many researchers.

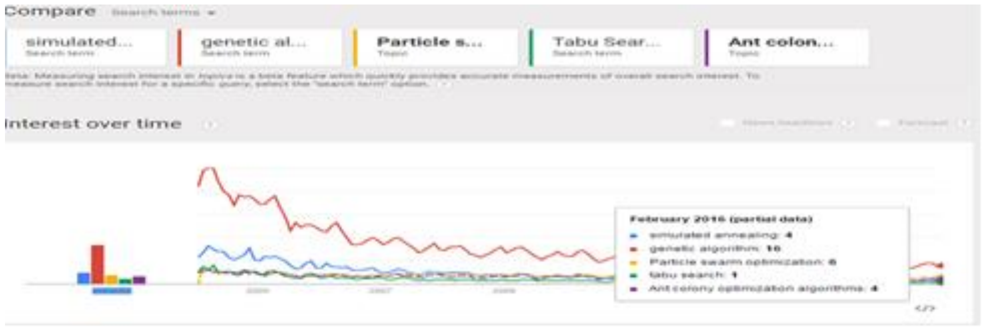

Fig.1. Popularity View of Different Algorithms using Google Trend

6. Conclusion

In this paper, we provide a multi-facet comparison between different types of metaheuristic and soft computing algorithms. These and other algorithms have been used for years by different researchers to solve various problems. Each algorithm has a different capability, purpose and strength to be able to obtain good results. In this review, we enlighten the readers on different aspects of the algorithm by comparing it with other algorithms. The literature is full of comparison between different algorithms in narrow-context problems. This review tries to establish a theoretical mapping of the comparison regardless of the problem context.

References

[1] Glover, F., Future Paths for Integer Programming and Links to Artificial Intelligence. Comput. Oper. Res., 1986. 13: p. 533–549.

[2] Zäpfel, G., R. Braune, and M. Bögl, Metaheuristics in General, in Metaheuristic Search Concepts. 2010, Springer Berlin Heidelberg. p. 67-73.

[3] Gendreau, M. and J.-Y. Potvin, Handbook of metaheuristics. Vol. 2. 2010: Springer.
[4] Talbi, E.-G., Metaheuristics: from design to implementation. 2009, Hoboken, N.J.: John Wiley & Sons.
[5] Yang, X.-S. and W.I.O. service), Engineering optimization an introduction with metaheuristic applications. 2010, Hoboken, N.J.: John Wiley.
[6] Hare, W., J. Nutini, and S. Tesfamariam, A Survey of Non-gradient Optimization Methods in Structural Engineering. Adv. Eng. Softw., 2013. 59: p. 19–28.
[7] Jin, Y., A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing, 2005. 9: p. 3-12.
[8] Davidor, Y. Epistasis Variance: A Viewpoint on GA-Hardness. in FOGA. 1990.
[9] Zäpfel, G., R. Braune, and M. Bögl, Search Heuristics, in Metaheuristic Search Concepts. 2010, Springer Berlin Heidelberg. p. 31-64.
[10] Tikk, D., L.T. Kóczy, and T.D. Gedeon, A survey on universal approximation and its limits in soft computing techniques. International Journal of Approximate Reasoning, 2003. 33: p. 185-202.
[11] Dorigo, M., Optimization, learning and natural algorithms. Ph. D. Thesis, Politecnico di Milano, Italy, 1992.
[12] Kennedy, J. and R. Eberhart. Particle swarm optimization. in Neural Networks, 1995. Proceedings., IEEE International Conference on. 1995.
[13] Alander, J.T., Population size, building blocks, fitness landscape and genetic algorithm search efficiency in combinatorial optimization: an empirical study. 1999.
[14] Kicinger, R., T. Arciszewski, and K.D. Jong, Evolutionary computation and structural design: A survey of the state-of-the-art. Computers & Structures, 2005. 83: p. 1943-1978.
[15] Deng, W., et al., A novel parallel hybrid intelligence optimization algorithm for a function approximation problem. Computers & Mathematics with Applications, 2012. 63: p. 325-336.
[16] GULSEN, M., A.E. SMITH, and D.M. TATE, A genetic algorithm approach to curve fitting. International Journal of Production Research, 1995. 33: p. 1911-1923.
[17] Roy, R., S. Hinduja, and R. Teti, Recent advances in engineering design optimisation: Challenges and future trends. CIRP Annals - Manufacturing Technology, 2008. 57: p. 697-715.
[18] Pedrycz, W. and F. Gomide, An introduction to fuzzy sets: analysis and design. 1998: Mit Press.
[19] Oduguwa, V., R. Roy, and D. Farrugia, Development of a soft computing-based framework for engineering design optimisation with quantitative and qualitative search spaces. Appl. Soft Comput., 2007. 7: p. 166–188.
[20] Mizukami, Y., Y. Wakasa, and K. Tanaka. A proposal of neural network architecture for non-linear function approximation. in Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004. 2004.
[21] Carpenter, W.C. and J.F. Barthelemy, Common Misconceptions about Neural Networks as Approximators. Journal of Computing in Civil Engineering, 1994. 8: p. 345-358.
[22] Crossley, M., A. Nisbet, and M. Amos, Quantifying the Impact of Parameter Tuning on Nature-Inspired Algorithms. 2013.
[23] Aarts, E., J. Korst, and W. Michiels, Simulated annealing, in Search methodologies. 2005, Springer. p. 187-210.
[24] Granville, V., M. Krivanek, and J.P. Rasson, Simulated annealing: a proof of convergence. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 1994. 16(6): p. 652-656.
[25] Dowsland, K.A. and J.M. Thompson, Simulated annealing, in Handbook of Natural Computing. 2012, Springer. p. 1623-1655.
[26] Berkhin, P., A survey of clustering data mining techniques, in Grouping multidimensional data. 2006, Springer. p. 25-71.
[27] Rui, X. and D. Wunsch, II, Survey of clustering algorithms. Neural Networks, IEEE Transactions on, 2005. 16(3): p. 645-678.
[28] Jain, A.K., Data clustering: 50 years beyond K-means. Pattern recognition letters, 2010. 31(8): p. 651-666.

[29] Digalakis, J.G. and K.G. Margaritis, on benchmarking functions for genetic algorithms. International Journal of Computer Mathematics, 2001. 77: p. 481-506.
[30] Yang, X.-S., 1 - Optimization and Metaheuristic Algorithms in Engineering, in Metaheuristics in Water, Geotechnical and Transport Engineering. 2013, Elsevier: Oxford. p. 1-23.
[31] Ying, K.-C., et al., A novel function approximation based on robust fuzzy regression algorithm model and particle swarm optimization. Applied Soft Computing, 2011. 11: p. 1820-1826.
[32] EL-Naggar, K.M., M.R. AlRashidi, and A.K. Al-Othman, Estimating the input–output parameters of thermal power plants using PSO. Energy Conversion and Management, 2009. 50: p. 1767-1772.
[33] Gutjahr, W.J., ACO algorithms with guaranteed convergence to the optimal solution. Information Processing Letters, 2002. 82(3): p. 145-153.
[34] Glover, F. and S.d. Hanafi, Tabu search and finite convergence. Discrete Applied Mathematics, 2002. 119(1–2): p. 3-36.
[35] NING Ai-ping, Z.X.-y., Convergence analysis of artificial bee colony algorithm. Control and Decision, 2013. 28(10): p. 1554-1558.
[36] Quan, Y. and G. Yin, Analyzing Convergence and Rates of Convergence of Particle Swarm Optimization Algorithms Using Stochastic Approximation Methods. Automatic Control, IEEE Transactions on, 2015. 60(7): p. 1760-1773.
[37] Real Life Applications of Soft Computing - CRC Press Book.
[38] Lee, K.S. and Z.W. Geem, A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Computer Methods in Applied Mechanics and Engineering, 2005. 194: p. 3902-3933.

Authors' Profiles

Abdulrahman Baqias obtained Ph.D. from King Fahd university of Petroleum and Minerals in 2016; he also obtained his Bachelor (Hons) in Software Engineering from Multimedia University, Malaysia in 2007 and his MSc from Staffordshire University, UK in 2010. His research interests including: software engineering, metaheuristics and search-based techniques and has published several articles in referred journal and conferences in these areas. He served as the session chair in IAENG conference in UK, 2013.

How to cite this paper: Abdulrahman Ahmed Bobakr Baqais,"A Multi-view Comparison of Various Metaheuristic and Soft Computing Algorithms", International Journal of Mathematical Sciences and Computing(IJMSC), Vol.3, No.4, pp.8-19, 2017.DOI: 10.5815/ijmsc.2017.04.02